Facsimile Transmission Form

To:

RECEIVED
CENTRAL FAX CENTER
DEC 14 2004

From:

Message: FAX

RECEIVED
CENTRAL FAX CENTER
DEC 14 2004

Docket No. JP919990286US1

Certification of Facsimile Transmission

Facsimile Number: | *703-872-9306*

Total Pages (Including this page): | 46

Date: *December 14, 2004*

I hereby certify that the following papers are being transmitted by facsimile device to the identified Facsimile Number in the Patent and Trademark Office on the date written above. The total number of pages transmitted (including this page) are listed above.

Saundra S. Christopher

Saundra S. Christopher

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Kawachiya et al.

Serial No.: 09/803,168

Filed: March 9, 2001 Group No.: 2126

For: Computer System, Memory Management Method, Storage Medium and Program Transmission Apparatus Examiner: Phuong Hoang

To: Commissioner of Patents
P. O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF
(PATENT APPLICATION-37 CFR 192)

Sir:

Transmitted herewith in triplicate is the APPEAL BRIEF in this application with respect to the Notice of Appeal filed October 14, 2004.

1. **STATUS OF APPLICATION**

This application is on behalf of

_X_ other than a small entity

___ small entity
verified statement: ___ attached ___ already filed

2. **FEE FOR FILING APPEAL BRIEF**

Pursuant to 37 CFR 1.17(f) the fee for filing the Appeal Brief is:

___ Small entity $250.00

X Other than a small entity $500.00

Appeal Brief fee due __$500.00___

3. **TOTAL FEE DUE**

The total fee due is:

Appeal brief fee ___$500.00_

Extension fee (if any) _____ TOTAL FEE DUE _____

4. **FEE PAYMENT**

___ Attached is a check in the sum of $ _____
_X_ Charge Account No. 09-0461 the sum of ___$500.00_____.
(a duplicate of this transmittal is attached)

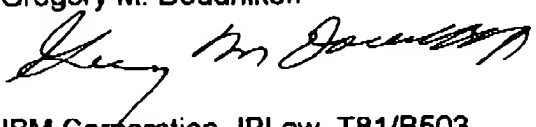
5. **FEE DEFICIENCY**

_X_ If any additional extension and/or fee is required, this is a request therefor and to charge Account No. 09-0461.

_X_ If any additional fee for claims is required, charge Account No. 09-0461.

Reg. No.: 32,847
Tel. No.: 919-254-1288

Gregory M. Doudnikoff



IBM Corporation, IPLaw, T81/B503
P.O. Box 12195
Research Triangle Park, NC 27709

Docket No. JP919990286US1

Certification of Facsimile Transmission

Facsimile Number: *703-872-9306*

Total Pages (including this page): 46

Date: *December 14, 2004*

I hereby certify that the following papers are being transmitted by facsimile device to the identified Facsimile Number in the Patent and Trademark Office on the date written above. The total number of pages transmitted (including this page) are listed above.

Saundra S. Christopher

Saundra S. Christopher

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Kawachiya et al.

Serial No.: 09/803,168

Filed: March 9, 2001

Group No.: 2126

For: Computer System, Memory Management Method, Storage Medium and Program Transmission Apparatus

Examiner: Phuong Hoang

To: Commissioner of Patents
P. O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF
(PATENT APPLICATION-37 CFR 192)

Sir:

Transmitted herewith in triplicate is the APPEAL BRIEF in this application with respect to the Notice of Appeal filed October 14, 2004.

1. **STATUS OF APPLICATION**

This application is on behalf of

_X_ other than a small entity

___ small entity
verified statement: ___ attached ___ already filed

2. **FEE FOR FILING APPEAL BRIEF**

Pursuant to 37 CFR 1.17(f) the fee for filing the Appeal Brief is:

___ Small entity $250.00

X Other than a small entity $500.00

Appeal Brief fee due __$500.00___

3. **TOTAL FEE DUE**

The total fee due is:

Appeal brief fee ___$500.00_

Extension fee (if any) _____

TOTAL FEE DUE _____

4. **FEE PAYMENT**

___ Attached is a check in the sum of $ _____
_X_ Charge Account No. 09-0461 the sum of ___$500.00_____.
(a duplicate of this transmittal is attached)

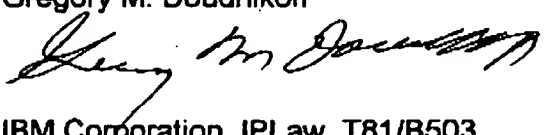
5. **FEE DEFICIENCY**

_X_ If any additional extension and/or fee is required, this is a request therefor and to charge Account No. 09-0461.

_X_ If any additional fee for claims is required, charge Account No. 09-0461.

Reg. No.: 32,847
Tel. No.: 919-254-1288

Gregory M. Doudnikoff



IBM Corporation, IPLaw, T81/B503
P.O. Box 12195
Research Triangle Park, NC 27709

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of	:	December 14, 2004
Kawachiya et al.	:	IBM Corporation
Ser. No. 09/803,168	:	Dept.T81/Bldg. 503
Filed: March 9, 2001	:	P.O. Box 12195
For: Computer System, Memory	:	Res. Tri. Park, NC 27709
Management Method, Storage Medium	:	Art Unit: 2126
and Program Transmission Apparatus	:	Examiner: Phuong Hoang

RECEIVED
CENTRAL FAX CENTER
DEC 14 2004

APPEAL BRIEF

Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

The following remarks in the Appeal for the above identified Application are respectfully submitted:

REAL PARTY IN INTEREST

This Application has been assigned to the International Business Machines Corporation.

RELATED APPEALS AND INTERFERENCES

Applicants know of no other Appeals or Interferences which will directly affect or be directly affected by or having a bearing on the Boards decision in the pending Serial No. 09/803,168

Appeal.

STATUS OF CLAIMS

The Application was originally filed with Claims 1 - 10. Claims 8-10 were amended in the Response dated March 29, 2004. Accordingly, Claims 1 - 10 remain pending, and these are the claims which are the subject of this Appeal. A copy of the appealed claims, Claims 1 - 10, is contained in the attached Appendix.

STATUS OF AMENDMENTS

Applicants' amendments contained in the Response filed on March 29, 2004 were entered. Applicants have made no further amendments.

SUMMARY OF THE INVENTION

The present invention provides a technique for skipping a locking process for an object in memory when a thread accesses an object that only it will access in order to reduce the load imposed on a system and to improve the overall system performance. A program executing in a computer system has multiple threads that share and access objects stored in memory. The objects have thread locality flags associated therewith that indicate the presence or absence of thread localities. The threads examine the thread locality flags for the objects they attempt to access to determine whether the corresponding objects, which are to be accessed, have localities for the threads. If, so the, threads skip the locking process and access objects immediately. If not, the object is locked prior to being accessed.

STATEMENT OF ISSUES PRESENTED

Applicants present for review the final rejection of Claim 1 under 35 U.S.C. 102(e) as being anticipated by US Patent Number 6,209,066 to Holzle et al (Holzle), of

PAGE 7/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

Claims 3, 4, 6, 7 and 9 as being anticipated by US Patent Number 5,862,376 to Steele, Jr. et al (Steele), of Claim 2 as being unpatentable over Holzle in view of US Patent Number 5,652,883 to Adcock, and of Claims 5, 8 and 10 as being unpatentable over Steele in view of Adcock.

GROUPING OF THE CLAIMS

Independent Claim 1 stands or falls alone.
Independent Claims 3, 6 and 9 stand or fall together.
Independent Claims 8 and 10 stand or fall together.
Dependent Claim 2 stands or falls alone.
Dependent Claims 4 and 7 stand or fall with Claims 3, 6 and 9.
Dependent Claim 5 stands or falls alone.

ARGUMENT

Applicants traverse the rejections below.

A. Independent Claim 1

Claim 1 was rejected as being anticipated by Holzle.

Holzle discloses a technique that provides thread-local "allocation area" for "fast-allocating" threads and omits locking if the allocation area is thread-local. In contrast with the present invention, Holzle's "thread-local" flag will not be changed after the allocation area is prepared.

In contrast, the present invention handles thread locality of dynamically changing data (or objects). The flag may be dynamically turned off during the execution.

Another difference between the present invention and Holzle is that the present

PAGE 8/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

invention is mainly targeting objects allocated from the allocation area, and not targeting the allocation area itself.

For example, Claim 1 recites "means for determining, when a thread attempts to access data, whether a specific thread indication is present relative to the data being accessed". Relative to this subject matter, private blocks 304b, 304d, 304f and the passage from Column 8, lines 1-5 are cited. This passage discusses what happens when thread 306a (a specific thread) attempts to **allocate a new object** in one of its assigned blocks (blocks already assigned to the thread 306a). There is no teaching or disclosure regarding determining, when a thread attempts to **access** data, whether a **specific thread indication is present** relative to the **data** being accessed.

The cited passage from Column 9, lines 1-20 does not discuss a technique for accessing data in which a determination is made as to whether or not a specific thread indication is present for the block being accessed. Rather, the passage describes a technique by which a thread allocates data to blocks

Since independent Claim 1 has been shown to patentably distinguish over Holzle art, it follows that the dependent Claim 2 also distinguishes over the cited art. Further differences between Claim 2 and the cited art will be discussed below.

Accordingly, Applicants submit that Claim 1 distinguishes over the cited art, and respectfully request that the Board overturn this rejection.

B. Claims 3, 4, 6, 7 and 9

Claims 3, 4, 6, 7 and 9 were rejected as being anticipated by Steele.

Steele includes the idea of omitting the recursive locks. The Steele system omits lock processing if the object is already locked ("previously synchronized" in their words) by the thread. This omission is performed irrelevantly to whether the object is accessible from other threads or not.

PAGE 9/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

The Steele "lock bit 226" does **not** mean that the object is inaccessible (unreachable) from other threads, so it is completely different from the thread-locality flag of the present invention. See the discussion of locking in Column 5, lines 35-45 of Steele.

Independent Claim 3 recites "flag data, provided for an object, for indicating an existence of a locality specifying that said object is to be accessed only by a specific thread". Relative to this subject matter, as noted above, the lock bit 226 is cited. This lock bit, "once obtained [by a thread] gives exclusive use (or sole access) to the object until the lock is released." See Column 4, lines 19-23. The Steele lock is employed recursively, and the lock is released when the thread is finished with the object.

Claim 3 also recites "means for having the specific thread access said object when said flag data for said object indicates said locality for said specific thread, without performing a locking process to reject access attempts by other threads or other objects before accessing said specific data". Relative to this subject matter, a passage from Column 4, lines 24-30 is cited. This passage states that the purpose of Steele is to provide a mechanism that manages the acquisition of object locks for each thread. There is no discussion therein of not performing a locking process to reject access attempts by other thread. When any one thread has possession of the lock on the object, all other threads that request possession of the lock on the object are forced to wait until all the earlier threads get and then release the lock on the object. See Column 5, lines 40 – 45. This would indicate that a locking is performed relative to the object and other threads that request the object.

Claim 3 also recites "means for having the specific thread perform said locking process before accessing said object when said flag data does not indicate said locality for said specific thread." Relative to this subject matter, the passage from Column 5 is once again cited. The Office Action appears to want to have the same subject matter from Steele disclose both the use of a lock bit that does not require a locking process

PAGE 10/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

and a locking process.

Since Claim 3 has been shown to patentably distinguish over Steele, it follows that dependent Claim 4 also distinguishes therefrom. And since independent Claims 6 and 9 were rejected for the same reasons as was Claims 3 and 4, it follows that Claim 6, its dependent Claim 7 and Claim 9 also distinguish over Steele.

Accordingly, Applicants submit that Claims 3, 4, 6, 7 and 9 further distinguish over the cited art, and respectfully request that the Board overturn this rejection.

C. Dependent Claim 2

Dependent Claim 2 was rejected over Holzle in view of Adcock.

Holzle discusses the locality of "allocation area", not the locality of each object. In contrast, the present invention describes the (dynamically changing) locality of each object allocated from the allocation area. Such a flag is not described in Holzle. Therefore, Applicant submits that dependent Claim 2 is not obvious at all and further distinguishes over the cited art.

Adcock shows an idea of generational garbage collection on a system with "conservative" stacks, which means that system cannot know whether each content of thread's stack is an object reference or not. The Adcock system needs to check stacks of **all** threads for garbage collection, as other previous garbage collections. In contrast, the present invention provides a new method for collecting memory area by just checking **one** thread's stack. Accordingly, Applicants submit that dependent Claim 2 further distinguishes over the cited art.

Accordingly, Applicants submit that Claim 2 distinguishes over the cited art, and respectfully request that the Board overtum this rejection.

PAGE 11/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

D. Claims 8 and 10

Independent Claims 8 and 10 (and dependent Claim 5) were rejected as being unpatentable over Holzle in view of Adcock.

Claim 8 recites "setting flag data indicating an existence of a locality indicating that a specific object that is created by a specific thread is to be accessed only by said specific thread". No subject matter is cited against this subject matter specifically in the Office Action. Applicants can find no teaching of an object being created by a specific thread in the cited art, much less the setting of a flag to indicate the existence of a locality for that created object so that the object is accessed only by the creating thread.

Claim 8 also recites "permitting said specific thread to detect an object for which the flag data indicates the existence of a locality for said specific thread and said specific thread does not have a reference pointer to said object". Relative to this subject matter (with respect to Claim 5), a passage from Column 5, lines 50-60 is apparently cited. This passage does not at all deal with the existence or lack of existence of a reference pointer to the object in the thread. Two passages from Adcock are also cited relative to this subject matter. But neither passage teaches, suggests or discloses that a thread has or does not have a reference pointer to an object.

As described above, Steele's "lock bit" is completely different from our thread-locality flag. As described above, Adcock's garbage collector needs to check all threads' stacks. Therefore, the subject matter provided in the subject claims is not rendered obvious by the combination of Steele and Adcock. Accordingly, Claims 5, 8 and 10 also patentably distinguish over the cited art.

Accordingly, Applicants submit that Claims 8 and 10 distinguish over the cited art, and respectfully request that the Board overturn this rejection.

PAGE 12/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

E. Claim 5

Dependent Claim 5 contains subject matter that was rejected for the same reasons as was the subject matter of Claims 8 and 10. Since Claims 8 and 10 have been shown above to distinguish over the cited art, it follows that Claim 5 also patentably distinguishes over the combination of Steele and Adcock.

Additionally, Claim 5 depends from independent Claim 3. Applicants have shown above that independent Claim 3 also distinguishes over the art cited against it. Accordingly, dependent Claim 3 also distinguishes over the art for this reason as well.

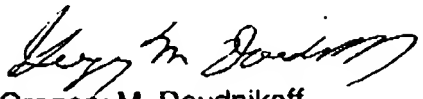
Accordingly, Applicants submit that Claim 5 distinguishes over the cited art, and respectfully request that the Board overturn this rejection.

SUMMARY

Applicants respectfully submit that the final rejection of the claims under 35 U.S.C. Sections 102 and 103 is improper and erroneous. Applicants respectfully urge the Board of Patent Appeals to reverse all grounds of the final rejection relative to the claims.

PAGE 13/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

Respectfully submitted,



Gregory M. Doudnikoff
Attorney for Applicant
Reg. No. 32,847

JP9-1999-0286
Docket No: ~~JA9-98-200~~
PHONE: 919-254-1288
FAX: 919-254-4330

PAGE 14/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

APPENDIX

1. A computer system having a data processing environment in which a program is divided into and executed as multiple threads, and in which said threads share and access data that is stored in a memory device, comprising:

means for indicating specific data that will be accessed only by a specific thread;

means for determining, when a thread attempts to access data, whether a specific thread indication is present relative to the data being accessed;

means for accessing said specific data without first performing a locking process to reject access attempts by other threads, when the specific thread indication is present; and

means for performing a locking process for the data being accessed before accessing the data when it is determined that no specific thread indication is present.

2. The computer system according to Claim 1, wherein said specific thread detects data, included in said data stored in said memory device, and said specific thread does not have a reference pointer to said data, and thereafter releases memory occupied by said data to provide storage space that is freely available.

3. In a data processing environment, a system in which multiple threads share and access objects, comprising:

flag data, provided for an object, for indicating an existence of a locality

specifying that said object is to be accessed only by a specific thread;

means for having the specific thread access said object when said flag data for said object indicates said locality for said specific thread, without performing a locking process to reject access attempts by other threads or other objects before accessing said specific data; and

means for having the specific thread perform said locking process before accessing said object when said flag data does not indicate said locality for said specific thread.

4. The computer system according to Claim 3, wherein, when said object is created by a thread, said object sets said flag data indicating a locality exists for said thread, and wherein, before said object is changed so that it can be accessed by another thread or another object, said locality indicated by said flag data is canceled.

5. The computer system according to Claim 3, wherein said specific thread detects an object for which said flag data indicates the existence of a locality for said specific thread but said specific thread does not have a reference pointer to said data, and thereafter releases said object to provide in a memory device storage space that is freely available.

6. A memory management method for a data processing environment in which a program is divided into and executed as multiple threads, and in which said threads share and access objects that are stored in a memory device, comprising the steps of:

setting flag data indicating an existence of a locality for a specific object that is

PAGE 16/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

created by a specific thread and that is to be accessed only by said specific thread;

canceling said locality indicated by said flag data before said specific object is changed so that said specific object can be accessed by another thread;

without performing a locking process to reject access attempts by other threads or objects, accessing said specific object when said flag data for said specific object indicates the existence of a locality for said specific thread; and

locking said specific object before accessing said specific object when there is no said flag data indicating a locality for said specific thread.

7. The memory management method according to Claim 6, wherein said step of canceling said locality indicated by said flag data for said specific object includes a step of:

performing said locking process, when said specific object has a locality for a specific thread, that was skipped at the time said specific object was accessed by said specific thread.

8. A memory management method for a data processing environment in which a program is divided into and is executed as multiple threads, and in which said threads share and access objects that are stored in a memory device, comprising the steps of:

setting flag data indicating an existence of a locality indicating that a specific object that is created by a specific thread is to be accessed only by said specific thread;

permitting said specific thread to detect an object for which the flag data

PAGE 17/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

indicates the existence of a locality for said specific thread and said specific thread does not have a reference pointer to said object; and

releasing said detected object to provide additional storage space in the memory device that may be freely used.

9. Computer readable code stored on computer readable medium or permitting a locking step to be skipped in certain situations relative to data in a multi-thread environment, comprising:

a process for setting flag data indicating the existence of a locality for a specific object that is created by a specific thread and that is to be accessed only by said specific thread;

a process for canceling said locality indicated by said flag data before said specific object is changed so that said specific object can be accessed by another thread;

a process accessing said specific object when said flag data for said specific object indicates the existence of a locality for said specific thread without performing a locking process to reject access attempts by other threads; and

a process for performing said locking process before accessing said specific object when said flag data indicates the absence of a locality for said specific thread.

10. Computer readable code stored on computer readable medium for performing memory management for a program that executes in multiple threads, comprising:

a process for setting flag data indicating existence of a locality indicating that a specific object that is created by a specific thread is to be accessed only by said

PAGE 18/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

specific thread;

a process for permitting said specific thread to detect an object for which flag data indicates the existence of a locality for said specific thread and said specific thread does not have a reference pointer to said object; and

a process for unlocking said detected object so that storage space may be freely used.

PAGE 19/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

RECEIVED
CENTRAL FAX CENTER
DEC 14 2004

In re application of	:	December 14, 2004
Kawachiya et al.	:	IBM Corporation
Ser. No. 09/803,168	:	Dept.T81/Bldg. 503
Filed: March 9, 2001	:	P.O. Box 12195
For: Computer System, Memory	:	Res. Tri. Park, NC 27709
Management Method, Storage Medium	:	Art Unit: 2126
and Program Transmission Apparatus	:	Examiner: Phuong Hoang

APPEAL BRIEF

Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

The following remarks in the Appeal for the above identified Application are respectfully submitted:

REAL PARTY IN INTEREST

This Application has been assigned to the International Business Machines Corporation.

RELATED APPEALS AND INTERFERENCES

Applicants know of no other Appeals or Interferences which will directly affect or be directly affected by or having a bearing on the Boards decision in the pending Serial No. 09/803,168

Appeal.

STATUS OF CLAIMS

The Application was originally filed with Claims 1 - 10. Claims 8-10 were amended in the Response dated March 29, 2004. Accordingly, Claims 1 - 10 remain pending, and these are the claims which are the subject of this Appeal. A copy of the appealed claims, Claims 1 - 10, is contained in the attached Appendix.

STATUS OF AMENDMENTS

Applicants' amendments contained in the Response filed on March 29, 2004 were entered. Applicants have made no further amendments.

SUMMARY OF THE INVENTION

The present invention provides a technique for skipping a locking process for an object in memory when a thread accesses an object that only it will access in order to reduce the load imposed on a system and to improve the overall system performance. A program executing in a computer system has multiple threads that share and access objects stored in memory. The objects have thread locality flags associated therewith that indicate the presence or absence of thread localities. The threads examine the thread locality flags for the objects they attempt to access to determine whether the corresponding objects, which are to be accessed, have localities for the threads. If, so the, threads skip the locking process and access objects immediately. If not, the object is locked prior to being accessed.

STATEMENT OF ISSUES PRESENTED

Applicants present for review the final rejection of Claim 1 under 35 U.S.C. 102(e) as being anticipated by US Patent Number 6,209,066 to Holzle et al (Holzle), of

PAGE 21/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

Claims 3, 4, 6, 7 and 9 as being anticipated by US Patent Number 5,862,376 to Steele, Jr. et al (Steele), of Claim 2 as being unpatentable over Holzle in view of US Patent Number 5,652,883 to Adcock, and of Claims 5, 8 and 10 as being unpatentable over Steele in view of Adcock.

GROUPING OF THE CLAIMS

Independent Claim 1 stands or falls alone.
Independent Claims 3, 6 and 9 stand or fall together.
Independent Claims 8 and 10 stand or fall together.
Dependent Claim 2 stands or falls alone.
Dependent Claims 4 and 7 stand or fall with Claims 3, 6 and 9.
Dependent Claim 5 stands or falls alone.

ARGUMENT

Applicants traverse the rejections below.

A. Independent Claim 1

Claim 1 was rejected as being anticipated by Holzle.

Holzle discloses a technique that provides thread-local "allocation area" for "fast-allocating" threads and omits locking if the allocation area is thread-local. In contrast with the present invention, Holzle's "thread-local" flag will not be changed after the allocation area is prepared.

In contrast, the present invention handles thread locality of dynamically changing data (or objects). The flag may be dynamically turned off during the execution.

Another difference between the present invention and Holzle is that the present

PAGE 22/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

invention is mainly targeting objects allocated from the allocation area, and not targeting the allocation area itself.

For example, Claim 1 recites "means for determining, when a thread attempts to access data, whether a specific thread indication is present relative to the data being accessed". Relative to this subject matter, private blocks 304b, 304d, 304f and the passage from Column 8, lines 1-5 are cited. This passage discusses what happens when thread 306a (a specific thread) attempts to **allocate a new object** in one of its assigned blocks (blocks already assigned to the thread 306a). There is no teaching or disclosure regarding determining, when a thread attempts to **access** data, whether a **specific thread indication is present** relative to the **data** being accessed.

The cited passage from Column 9, lines 1-20 does not discuss a technique for accessing data in which a determination is made as to whether or not a specific thread indication is present for the block being accessed. Rather, the passage describes a technique by which a thread allocates data to blocks

Since independent Claim 1 has been shown to patentably distinguish over Holzle art, it follows that the dependent Claim 2 also distinguishes over the cited art. Further differences between Claim 2 and the cited art will be discussed below.

Accordingly, Applicants submit that Claim 1 distinguishes over the cited art, and respectfully request that the Board overturn this rejection.

B. Claims 3, 4, 6, 7 and 9

Claims 3, 4, 6, 7 and 9 were rejected as being anticipated by Steele.

Steele includes the idea of omitting the recursive locks. The Steele system omits lock processing if the object is already locked ("previously synchronized" in their words) by the thread. This omission is performed irrelevantly to whether the object is accessible from other threads or not.

PAGE 23/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

The Steele "lock bit 226" does **not** mean that the object is inaccessible (unreachable) from other threads, so it is completely different from the thread-locality flag of the present invention. See the discussion of locking in Column 5, lines 35-45 of Steele.

Independent Claim 3 recites "flag data, provided for an object, for indicating an existence of a locality specifying that said object is to be accessed only by a specific thread". Relative to this subject matter, as noted above, the lock bit 226 is cited. This lock bit, "once obtained [by a thread] gives exclusive use (or sole access) to the object until the lock is released." See Column 4, lines 19-23. The Steele lock is employed recursively, and the lock is released when the thread is finished with the object.

Claim 3 also recites "means for having the specific thread access said object when said flag data for said object indicates said locality for said specific thread, without performing a locking process to reject access attempts by other threads or other objects before accessing said specific data". Relative to this subject matter, a passage from Column 4, lines 24-30 is cited. This passage states that the purpose of Steele is to provide a mechanism that manages the acquisition of object locks for each thread. There is no discussion therein of not performing a locking process to reject access attempts by other thread. When any one thread has possession of the lock on the object, all other threads that request possession of the lock on the object are forced to wait until all the earlier threads get and then release the lock on the object. See Column 5, lines 40 – 45. This would indicate that a locking is performed relative to the object and other threads that request the object.

Claim 3 also recites "means for having the specific thread perform said locking process before accessing said object when said flag data does not indicate said locality for said specific thread." Relative to this subject matter, the passage from Column 5 is once again cited. The Office Action appears to want to have the same subject matter from Steele disclose both the use of a lock bit that does not require a locking process

PAGE 24/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

and a locking process.

Since Claim 3 has been shown to patentably distinguish over Steele, it follows that dependent Claim 4 also distinguishes therefrom. And since independent Claims 6 and 9 were rejected for the same reasons as was Claims 3 and 4, it follows that Claim 6, its dependent Claim 7 and Claim 9 also distinguish over Steele.

Accordingly, Applicants submit that Claims 3, 4, 6, 7 and 9 further distinguish over the cited art, and respectfully request that the Board overturn this rejection.

C. Dependent Claim 2

Dependent Claim 2 was rejected over Holzle in view of Adcock.

Holzle discusses the locality of "allocation area", not the locality of each object. In contrast, the present invention describes the (dynamically changing) locality of each object allocated from the allocation area. Such a flag is not described in Holzle. Therefore, Applicant submits that dependent Claim 2 is not obvious at all and further distinguishes over the cited art.

Adcock shows an idea of generational garbage collection on a system with "conservative" stacks, which means that system cannot know whether each content of thread's stack is an object reference or not. The Adcock system needs to check stacks of **all** threads for garbage collection, as other previous garbage collections. In contrast, the present invention provides a new method for collecting memory area by just checking **one** thread's stack. Accordingly, Applicants submit that dependent Claim 2 further distinguishes over the cited art.

Accordingly, Applicants submit that Claim 2 distinguishes over the cited art, and respectfully request that the Board overturn this rejection.

PAGE 25/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

D. Claims 8 and 10

Independent Claims 8 and 10 (and dependent Claim 5) were rejected as being unpatentable over Holzle in view of Adcock.

Claim 8 recites "setting flag data indicating an existence of a locality indicating that a specific object that is created by a specific thread is to be accessed only by said specific thread". No subject matter is cited against this subject matter specifically in the Office Action. Applicants can find no teaching of an object being created by a specific thread in the cited art, much less the setting of a flag to indicate the existence of a locality for that created object so that the object is accessed only by the creating thread.

Claim 8 also recites "permitting said specific thread to detect an object for which the flag data indicates the existence of a locality for said specific thread and said specific thread does not have a reference pointer to said object". Relative to this subject matter (with respect to Claim 5), a passage from Column 5, lines 50-60 is apparently cited. This passage does not at all deal with the existence or lack of existence of a reference pointer to the object in the thread. Two passages from Adcock are also cited relative to this subject matter. But neither passage teaches, suggests or discloses that a thread has or does not have a reference pointer to an object.

As described above, Steele's "lock bit" is completely different from our thread-locality flag. As described above, Adcock's garbage collector needs to check all threads' stacks. Therefore, the subject matter provided in the subject claims is not rendered obvious by the combination of Steele and Adcock. Accordingly, Claims 5, 8 and 10 also patentably distinguish over the cited art.

Accordingly, Applicants submit that Claims 8 and 10 distinguish over the cited art, and respectfully request that the Board overturn this rejection.

PAGE 26/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

E. Claim 5

Dependent Claim 5 contains subject matter that was rejected for the same reasons as was the subject matter of Claims 8 and 10. Since Claims 8 and 10 have been shown above to distinguish over the cited art, it follows that Claim 5 also patentably distingulshes over the combination of Steele and Adcock.

Additionally, Claim 5 depends from independent Claim 3. Applicants have shown above that independent Claim 3 also distinguishes over the art cited against it. Accordingly, dependent Claim 3 also distinguishes over the art for this reason as well.

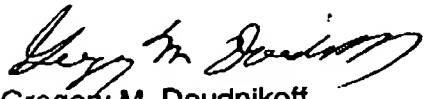
Accordingly, Applicants submit that Claim 5 distinguishes over the cited art, and respectfully request that the Board overturn this rejection.

SUMMARY

Applicants respectfully submit that the final rejection of the claims under 35 U.S.C. Sections 102 and 103 is improper and erroneous. Applicants respectfully urge the Board of Patent Appeals to reverse all grounds of the final rejection relative to the claims.

PAGE 27/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

Respectfully submitted,



Gregory M. Doudnikoff
Attorney for Applicant
Reg. No. 32,847

Docket No: ~~JAS-98-200~~ JP9-1999-0286
PHONE: 919-254-1288
FAX: 919-254-4330

PAGE 28/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

APPENDIX

1. A computer system having a data processing environment in which a program is divided into and executed as multiple threads, and in which said threads share and access data that is stored in a memory device, comprising:

means for indicating specific data that will be accessed only by a specific thread;

means for determining, when a thread attempts to access data, whether a specific thread indication is present relative to the data being accessed;

means for accessing said specific data without first performing a locking process to reject access attempts by other threads, when the specific thread indication is present; and

means for performing a locking process for the data being accessed before accessing the data when it is determined that no specific thread indication is present.

2. The computer system according to Claim 1, wherein said specific thread detects data, included in said data stored in said memory device, and said specific thread does not have a reference pointer to said data, and thereafter releases memory occupied by said data to provide storage space that is freely available.

3. In a data processing environment, a system in which multiple threads share and access objects, comprising:

flag data, provided for an object, for indicating an existence of a locality

specifying that said object is to be accessed only by a specific thread;

means for having the specific thread access said object when said flag data for said object indicates said locality for said specific thread, without performing a locking process to reject access attempts by other threads or other objects before accessing said specific data; and

means for having the specific thread perform said locking process before accessing said object when said flag data does not indicate said locality for said specific thread.

4. The computer system according to Claim 3, wherein, when said object is created by a thread, said object sets said flag data indicating a locality exists for said thread, and wherein, before said object is changed so that it can be accessed by another thread or another object, said locality indicated by said flag data is canceled.

5. The computer system according to Claim 3, wherein said specific thread detects an object for which said flag data indicates the existence of a locality for said specific thread but said specific thread does not have a reference pointer to said data, and thereafter releases said object to provide in a memory device storage space that is freely available.

6. A memory management method for a data processing environment in which a program is divided into and executed as multiple threads, and in which said threads share and access objects that are stored in a memory device, comprising the steps of:

setting flag data indicating an existence of a locality for a specific object that is

PAGE 30/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

created by a specific thread and that is to be accessed only by said specific thread;

canceling said locality indicated by said flag data before said specific object is changed so that said specific object can be accessed by another thread;

without performing a locking process to reject access attempts by other threads or objects, accessing said specific object when said flag data for said specific object indicates the existence of a locality for said specific thread; and

locking said specific object before accessing said specific object when there is no said flag data indicating a locality for said specific thread.

7. The memory management method according to Claim 6, wherein said step of canceling said locality indicated by said flag data for said specific object includes a step of:

performing said locking process, when said specific object has a locality for a specific thread, that was skipped at the time said specific object was accessed by said specific thread.

8. A memory management method for a data processing environment in which a program is divided into and is executed as multiple threads, and in which said threads share and access objects that are stored in a memory device, comprising the steps of:

setting flag data indicating an existence of a locality indicating that a specific object that is created by a specific thread is to be accessed only by said specific thread;

permitting said specific thread to detect an object for which the flag data

PAGE 31/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

indicates the existence of a locality for said specific thread and said specific thread does not have a reference pointer to said object; and

releasing said detected object to provide additional storage space in the memory device that may be freely used.

9. Computer readable code stored on computer readable medium or permitting a locking step to be skipped in certain situations relative to data in a multi-thread environment, comprising:

a process for setting flag data indicating the existence of a locality for a specific object that is created by a specific thread and that is to be accessed only by said specific thread;

a process for canceling said locality indicated by said flag data before said specific object is changed so that said specific object can be accessed by another thread;

a process accessing said specific object when said flag data for said specific object indicates the existence of a locality for said specific thread without performing a locking process to reject access attempts by other threads; and

a process for performing said locking process before accessing said specific object when said flag data indicates the absence of a locality for said specific thread.

10. Computer readable code stored on computer readable medium for performing memory management for a program that executes in multiple threads, comprising:

a process for setting flag data indicating existence of a locality indicating that a specific object that is created by a specific thread is to be accessed only by said

PAGE 32/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

specific thread;

a process for permitting said specific thread to detect an object for which flag data indicates the existence of a locality for said specific thread and said specific thread does not have a reference pointer to said object; and

a process for unlocking said detected object so that storage space may be freely used.

*PAGE 33/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of	:	December 14, 2004
Kawachiya et al.	:	IBM Corporation
Ser. No. 09/803,168	:	Dept.T81/Bldg. 503
Filed: March 9, 2001	:	P.O. Box 12195
For: Computer System, Memory	:	Res. Tri. Park, NC 27709
Management Method, Storage Medium	:	Art Unit: 2126
and Program Transmission Apparatus	:	Examiner: Phuong Hoang

RECEIVED
CENTRAL FAX CENTER
DEC 14 2004

APPEAL BRIEF

Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

The following remarks in the Appeal for the above identified Application are respectfully submitted:

REAL PARTY IN INTEREST

This Application has been assigned to the International Business Machines Corporation.

RELATED APPEALS AND INTERFERENCES

Applicants know of no other Appeals or Interferences which will directly affect or be directly affected by or having a bearing on the Boards decision in the pending Serial No. 09/803,168

Appeal.

STATUS OF CLAIMS

The Application was originally filed with Claims 1 - 10. Claims 8-10 were amended in the Response dated March 29, 2004. Accordingly, Claims 1 - 10 remain pending, and these are the claims which are the subject of this Appeal. A copy of the appealed claims, Claims 1 - 10, is contained in the attached Appendix.

STATUS OF AMENDMENTS

Applicants' amendments contained in the Response filed on March 29, 2004 were entered. Applicants have made no further amendments.

SUMMARY OF THE INVENTION

The present invention provides a technique for skipping a locking process for an object in memory when a thread accesses an object that only it will access in order to reduce the load imposed on a system and to improve the overall system performance. A program executing in a computer system has multiple threads that share and access objects stored in memory. The objects have thread locality flags associated therewith that indicate the presence or absence of thread localities. The threads examine the thread locality flags for the objects they attempt to access to determine whether the corresponding objects, which are to be accessed, have localities for the threads. If, so the, threads skip the locking process and access objects immediately. If not, the object is locked prior to being accessed.

STATEMENT OF ISSUES PRESENTED

Applicants present for review the final rejection of Claim 1 under 35 U.S.C. 102(e) as being anticipated by US Patent Number 6,209,066 to Holzle et al (Holzle), of

PAGE 35/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

Claims 3, 4, 6, 7 and 9 as being anticipated by US Patent Number 5,862,376 to Steele, Jr. et al (Steele), of Claim 2 as being unpatentable over Holzle in view of US Patent Number 5,652,883 to Adcock, and of Claims 5, 8 and 10 as being unpatentable over Steele in view of Adcock.

GROUPING OF THE CLAIMS

Independent Claim 1 stands or falls alone.
Independent Claims 3, 6 and 9 stand or fall together.
Independent Claims 8 and 10 stand or fall together.
Dependent Claim 2 stands or falls alone.
Dependent Claims 4 and 7 stand or fall with Claims 3, 6 and 9.
Dependent Claim 5 stands or falls alone.

ARGUMENT

Applicants traverse the rejections below.

A. Independent Claim 1

Claim 1 was rejected as being anticipated by Holzle.

Holzle discloses a technique that provides thread-local "allocation area" for "fast-allocating" threads and omits locking if the allocation area is thread-local. In contrast with the present invention, Holzle's "thread-local" flag will not be changed after the allocation area is prepared.

In contrast, the present invention handles thread locality of dynamically changing data (or objects). The flag may be dynamically turned off during the execution.

Another difference between the present invention and Holzle is that the present

PAGE 36/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

invention is mainly targeting objects allocated from the allocation area, and not targeting the allocation area itself.

For example, Claim 1 recites "means for determining, when a thread attempts to access data, whether a specific thread indication is present relative to the data being accessed". Relative to this subject matter, private blocks 304b, 304d, 304f and the passage from Column 8, lines 1-5 are cited. This passage discusses what happens when thread 306a (a specific thread) attempts to **allocate a new object** in one of its assigned blocks (blocks already assigned to the thread 306a). There is no teaching or disclosure regarding determining, when a thread attempts to **access** data, whether a **specific thread indication is present** relative to the **data** being accessed.

The cited passage from Column 9, lines 1-20 does not discuss a technique for accessing data in which a determination is made as to whether or not a specific thread indication is present for the block being accessed. Rather, the passage describes a technique by which a thread allocates data to blocks

Since independent Claim 1 has been shown to patentably distinguish over Holzle art, it follows that the dependent Claim 2 also distinguishes over the cited art. Further differences between Claim 2 and the cited art will be discussed below.

Accordingly, Applicants submit that Claim 1 distinguishes over the cited art, and respectfully request that the Board overturn this rejection.

B. Claims 3, 4, 6, 7 and 9

Claims 3, 4, 6, 7 and 9 were rejected as being anticipated by Steele.

Steele includes the idea of omitting the recursive locks. The Steele system omits lock processing if the object is already locked ("previously synchronized" in their words) by the thread. This omission is performed irrelevantly to whether the object is accessible from other threads or not.

PAGE 37/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

The Steele "lock bit 226" does **not** mean that the object is inaccessible (unreachable) from other threads, so it is completely different from the thread-locality flag of the present invention. See the discussion of locking in Column 5, lines 35-45 of Steele.

Independent Claim 3 recites "flag data, provided for an object, for indicating an existence of a locality specifying that said object is to be accessed only by a specific thread". Relative to this subject matter, as noted above, the lock bit 226 is cited. This lock bit, "once obtained [by a thread] gives exclusive use (or sole access) to the object until the lock is released." See Column 4, lines 19-23. The Steele lock is employed recursively, and the lock is released when the thread is finished with the object.

Claim 3 also recites "means for having the specific thread access said object when said flag data for said object indicates said locality for said specific thread, without performing a locking process to reject access attempts by other threads or other objects before accessing said specific data". Relative to this subject matter, a passage from Column 4, lines 24-30 is cited. This passage states that the purpose of Steele is to provide a mechanism that manages the acquisition of object locks for each thread. There is no discussion therein of not performing a locking process to reject access attempts by other thread. When any one thread has possession of the lock on the object, all other threads that request possession of the lock on the object are forced to wait until all the earlier threads get and then release the lock on the object. See Column 5, lines 40 – 45. This would indicate that a locking is performed relative to the object and other threads that request the object.

Claim 3 also recites "means for having the specific thread perform said locking process before accessing said object when said flag data does not indicate said locality for said specific thread." Relative to this subject matter, the passage from Column 5 is once again cited. The Office Action appears to want to have the same subject matter from Steele disclose both the use of a lock bit that does not require a locking process

PAGE 38/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

and a locking process.

Since Claim 3 has been shown to patentably distinguish over Steele, it follows that dependent Claim 4 also distinguishes therefrom. And since independent Claims 6 and 9 were rejected for the same reasons as was Claims 3 and 4, it follows that Claim 6, its dependent Claim 7 and Claim 9 also distinguish over Steele.

Accordingly, Applicants submit that Claims 3, 4, 6, 7 and 9 further distinguish over the cited art, and respectfully request that the Board overturn this rejection.

C. Dependent Claim 2

Dependent Claim 2 was rejected over Holzle in view of Adcock.

Holzle discusses the locality of "allocation area", not the locality of each object. In contrast, the present invention describes the (dynamically changing) locality of each object allocated from the allocation area. Such a flag is not described in Holzle. Therefore, Applicant submits that dependent Claim 2 is not obvious at all and further distinguishes over the cited art.

Adcock shows an idea of generational garbage collection on a system with "conservative" stacks, which means that system cannot know whether each content of thread's stack is an object reference or not. The Adcock system needs to check stacks of all threads for garbage collection, as other previous garbage collections. In contrast, the present invention provides a new method for collecting memory area by just checking **one** thread's stack. Accordingly, Applicants submit that dependent Claim 2 further distinguishes over the cited art.

Accordingly, Applicants submit that Claim 2 distinguishes over the cited art, and respectfully request that the Board overturn this rejection.

-PAGE 39/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

D. Claims 8 and 10

Independent Claims 8 and 10 (and dependent Claim 5) were rejected as being unpatentable over Holzle in view of Adcock.

Claim 8 recites "setting flag data indicating an existence of a locality indicating that a specific object that is created by a specific thread is to be accessed only by said specific thread". No subject matter is cited against this subject matter specifically in the Office Action. Applicants can find no teaching of an object being created by a specific thread in the cited art, much less the setting of a flag to indicate the existence of a locality for that created object so that the object is accessed only by the creating thread.

Claim 8 also recites "permitting said specific thread to detect an object for which the flag data indicates the existence of a locality for said specific thread and said specific thread does not have a reference pointer to said object". Relative to this subject matter (with respect to Claim 5), a passage from Column 5, lines 50-60 is apparently cited. This passage does not at all deal with the existence or lack of existence of a reference pointer to the object in the thread. Two passages from Adcock are also cited relative to this subject matter. But neither passage teaches, suggests or discloses that a thread has or does not have a reference pointer to an object.

As described above, Steele's "lock bit" is completely different from our thread-locality flag. As described above, Adcock's garbage collector needs to check all threads' stacks. Therefore, the subject matter provided in the subject claims is not rendered obvious by the combination of Steele and Adcock. Accordingly, Claims 5, 8 and 10 also patentably distinguish over the cited art.

Accordingly, Applicants submit that Claims 8 and 10 distinguish over the cited art, and respectfully request that the Board overturn this rejection.

PAGE 40/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

E. Claim 5

Dependent Claim 5 contains subject matter that was rejected for the same reasons as was the subject matter of Claims 8 and 10. Since Claims 8 and 10 have been shown above to distinguish over the cited art, it follows that Claim 5 also patentably distinguishes over the combination of Steele and Adcock.

Additionally, Claim 5 depends from independent Claim 3. Applicants have shown above that independent Claim 3 also distinguishes over the art cited against it. Accordingly, dependent Claim 3 also distinguishes over the art for this reason as well.

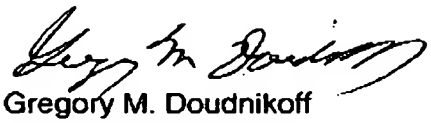
Accordingly, Applicants submit that Claim 5 distinguishes over the cited art, and respectfully request that the Board overturn this rejection.

SUMMARY

Applicants respectfully submit that the final rejection of the claims under 35 U.S.C. Sections 102 and 103 is improper and erroneous. Applicants respectfully urge the Board of Patent Appeals to reverse all grounds of the final rejection relative to the claims.

-PAGE 41/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

Respectfully submitted,



Gregory M. Doudnikoff
Attorney for Applicant
Reg. No. 32,847

JP9-1999-0286
Docket No: ~~JA9-98-200~~
PHONE: 919-254-1288
FAX: 919-254-4330

PAGE 42/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

APPENDIX

1. A computer system having a data processing environment in which a program is divided into and executed as multiple threads, and in which said threads share and access data that is stored in a memory device, comprising:

means for indicating specific data that will be accessed only by a specific thread;

means for determining, when a thread attempts to access data, whether a specific thread indication is present relative to the data being accessed;

means for accessing said specific data without first performing a locking process to reject access attempts by other threads, when the specific thread indication is present; and

means for performing a locking process for the data being accessed before accessing the data when it is determined that no specific thread indication is present.

2. The computer system according to Claim 1, wherein said specific thread detects data, included in said data stored in said memory device, and said specific thread does not have a reference pointer to said data, and thereafter releases memory occupied by said data to provide storage space that is freely available.

3. In a data processing environment, a system in which multiple threads share and access objects, comprising:

flag data, provided for an object, for indicating an existence of a locality

specifying that said object is to be accessed only by a specific thread;

means for having the specific thread access said object when said flag data for said object indicates said locality for said specific thread, without performing a locking process to reject access attempts by other threads or other objects before accessing said specific data; and

means for having the specific thread perform said locking process before accessing said object when said flag data does not indicate said locality for said specific thread.

4. The computer system according to Claim 3, wherein, when said object is created by a thread, said object sets said flag data indicating a locality exists for said thread, and wherein, before said object is changed so that it can be accessed by another thread or another object, said locality indicated by said flag data is canceled.

5. The computer system according to Claim 3, wherein said specific thread detects an object for which said flag data indicates the existence of a locality for said specific thread but said specific thread does not have a reference pointer to said data, and thereafter releases said object to provide in a memory device storage space that is freely available.

6. A memory management method for a data processing environment in which a program is divided into and executed as multiple threads, and in which said threads share and access objects that are stored in a memory device, comprising the steps of:

setting flag data indicating an existence of a locality for a specific object that is

PAGE 44/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

created by a specific thread and that is to be accessed only by said specific thread;

canceling said locality indicated by said flag data before said specific object is changed so that said specific object can be accessed by another thread;

without performing a locking process to reject access attempts by other threads or objects, accessing said specific object when said flag data for said specific object indicates the existence of a locality for said specific thread; and

locking said specific object before accessing said specific object when there is no said flag data indicating a locality for said specific thread.

7. The memory management method according to Claim 6, wherein said step of canceling said locality indicated by said flag data for said specific object includes a step of:

performing said locking process, when said specific object has a locality for a specific thread, that was skipped at the time said specific object was accessed by said specific thread.

8. A memory management method for a data processing environment in which a program is divided into and is executed as multiple threads, and in which said threads share and access objects that are stored in a memory device, comprising the steps of:

setting flag data indicating an existence of a locality indicating that a specific object that is created by a specific thread is to be accessed only by said specific thread;

permitting said specific thread to detect an object for which the flag data

—PAGE 45/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

indicates the existence of a locality for said specific thread and said specific thread does not have a reference pointer to said object; and

releasing said detected object to provide additional storage space in the memory device that may be freely used.

9. Computer readable code stored on computer readable medium or permitting a locking step to be skipped in certain situations relative to data in a multi-thread environment, comprising:

a process for setting flag data indicating the existence of a locality for a specific object that is created by a specific thread and that is to be accessed only by said specific thread;

a process for canceling said locality indicated by said flag data before said specific object is changed so that said specific object can be accessed by another thread;

a process accessing said specific object when said flag data for said specific object indicates the existence of a locality for said specific thread without performing a locking process to reject access attempts by other threads; and

a process for performing said locking process before accessing said specific object when said flag data indicates the absence of a locality for said specific thread.

10. Computer readable code stored on computer readable medium for performing memory management for a program that executes in multiple threads, comprising:

a process for setting flag data indicating existence of a locality indicating that a specific object that is created by a specific thread is to be accessed only by said

—PAGE 46/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14

specific thread;

a process for permitting said specific thread to detect an object for which flag data indicates the existence of a locality for said specific thread and said specific thread does not have a reference pointer to said object; and

a process for unlocking said detected object so that storage space may be freely used.

-PAGE 47/47 * RCVD AT 12/14/2004 3:47:48 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/2 * DNIS:8729306 * CSID:919-254-4330 * DURATION (mm-ss):10-14